

AD-A054 189

NAVAL RESEARCH LAB WASHINGTON D C  
AN EXPERIMENT IN DATABASE ACCESS CONTROL.(U)  
MAR 78 F A MANOLA, D K HSIAO  
NRL-8176

F/G 9/2

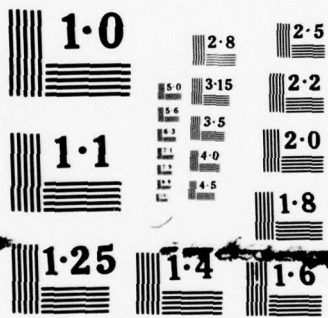
UNCLASSIFIED

NL

1 OF 1  
ADA  
054189



END  
DATE  
FILMED  
6-78  
DDC



NATIONAL BUREAU OF STANDARDS  
MICROCOPY RESOLUTION TEST CHART

FOR FURTHER TRAN

NRL Report 8176

AD A 054189

## An Experiment In Database Access Control

FRANK A. MANOLA

*Information Systems Staff  
Communication Sciences Division*

DAVID K. HSIAO

*The Ohio State University*

March 16, 1978



AD No. \_\_\_\_\_  
DDC FILE COPY



NAVAL RESEARCH LABORATORY  
Washington, D.C.

Approved for public release; distribution unlimited.

(14) NRL-8176

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER NRL Report 8176 ✓	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) (6) AN EXPERIMENT IN DATABASE ACCESS CONTROL	5. TYPE OF REPORT & PERIOD COVERED (7) Interim report, continuing NRL problem	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) (10) Frank A. Manola, David K. Hsiao	8. CONTRACT OR GRANT NUMBER(s) (11) 16 Mar 78	
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Research Laboratory Washington, D.C. 20375 (16) F21211	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS NRL Problem B02-24 Projects RF-21-211-401 and FR-22242-401	
11. CONTROLLING OFFICE NAME AND ADDRESS Department of the Navy Office of Naval Research Arlington, VA 22217 (17) RF21211401	12. REPORT DATE March 16, 1978 (12) 38p	
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	13. NUMBER OF PAGES 37	15. SECURITY CLASS. (of this report) Unclassified
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES A version of this report was presented at the First International Conference on Computer Software and Applications (COMPSAC 77), Nov 8-11, Chicago, Illinois. NTIS WGA Category 62		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Database Database management Data security Access control Command and control		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) To prevent unauthorized access to sensitive information, contemporary data-management systems may require redundant hardware, software, and data. In a military environment, where information is classified and users must have specific access clearances, the lack of automated enforcement of administrative control and the tendency toward overclassification can further increase hardware, software, and data requirements. A naval database was created for an experimental data-secure system to show that the system's advanced features can help solve real-world access-control problems and eliminate related hardware, software and data redundancy. (Continued)		

DD FORM 1 JAN 73 1473

EDITION OF 1 NOV 65 IS OBSOLETE  
S/N 0102-LF-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

251 950

AW

ABSTRACT: (Cont'd)

Typical scenarios which characterize multilevel access-control requirements in a real military environment were devised and tested on the system. The experience from demonstration of the system relates to the general area of access control in database-management systems.

A

ACCESS FOR	
NTIS	Table Section <input checked="" type="checkbox"/>
DOC	B II Section <input type="checkbox"/>
UNCLASSIFIED	<input type="checkbox"/>
DISTRIBUTION/AVAILABILITY CODES	
Dis	SPECIAL
A	

## CONTENTS

INTRODUCTION AND BACKGROUND .....	1
A REAL-WORLD DATABASE APPLICATION .....	2
Overall Access-Control Problems .....	2
Concepts and Terminology .....	5
Database-Management Capabilities Required for Contact Processing .....	6
AN EXPERIMENTAL SOLUTION TO THE DATABASE APPLICATION .....	7
Basic Elements of the Experimental Solution .....	8
The Database Organization .....	9
DEMONSTRATION OF THE SOLUTION .....	14
Access-Control Capabilities Demonstrated .....	14
Demonstration Plan .....	15
RESULTS OF THE EXPERIMENT .....	17
Data-Structure Transformation .....	17
Relationship of Security and Integrity Constraints .....	17
Usefulness of Value-Based Access Control .....	18
COMMENTS ON DEMONSTRATIONS OF DATABASE SECURITY MECHANISMS .....	18
Factors in Providing a Secure Database Facility .....	18
The Security-Kernel Approach .....	19
External System Characteristics .....	20
Effect of These Factors on the Experiment .....	20
CONCLUDING REMARKS .....	21
ACKNOWLEDGMENTS .....	21
REFERENCES .....	21
APPENDIX — Examples of HSDMS Usage Corresponding to Various Steps in the Demonstration Plan .....	22

## AN EXPERIMENT IN DATABASE ACCESS CONTROL

### INTRODUCTION AND BACKGROUND

In recent years, the subject of access control in database-management systems has been receiving attention in both research and user communities. This interest may in part be due to the influence of related research in the area of operating-system access control and in part due to the increasing user awareness of the operational risks of database-management systems that lack adequate access control. Consequently, a number of research papers and reports have described various access-control features for database-management systems. However, in many cases the proposed access-control features have not been accompanied by examples of realistic situations in which such features would be useful. This report attempts to take a different approach. By describing how the advanced access-control facilities of an experimental database-management system could be used to solve the access-control problems associated with a real-world database application, we hope to show the usefulness of these access-control facilities in solving real-world problems.

In this report we describe an existing Navy database application and the access-control problems associated with it. We show how we *emulated the application on an experimental database-management system with enhanced access-control facilities*. This emulation included an analysis of the application, characterization of an existing Navy database system which supported the application, generation of a new database, and loading of the database into the experimental system. We then compare the characteristics of the existing Navy database system for the application with the characteristics of the experimental database system for the same application. This comparison is intended to provide specific examples of how enhanced access-control facilities would be useful for that application. Because many of the properties of the chosen application are typical of other database applications, we feel that the demonstrated advantages of access-control facilities are applicable to a wide range of real-world database-access-control problems.

In this report we also describe a demonstration of the experimental system. Finally, we attempt to assess the usefulness of the experiment and try to relate our experience with this single Navy database application to that which might be expected in general with similar access-control mechanisms.

The experimental database-management system, the Highly Secure Database Management System (HSDMS), was developed at the Ohio State University as a part of a continuing research program in the area of data security and access control [1-6]. The experiment was conducted by the Naval Research Laboratory (NRL) and the Ohio State University (OSU), in cooperation with the Office of Naval Research (ONR), the Information Systems Office of the

Commander-in-Chief, Atlantic Fleet (CINCLANTFLT), the Naval Electronic Systems Command, and the Navy Regional Data Automation Center (then the Naval Command Systems Support Activity). Early in 1975, ONR, NRL, and CINCLANTFLT agreed that CINCLANTFLT would suggest a realistic Navy application illustrating access-control problems for an experiment on HSDMS. Acting as an intermediary between CINCLANTFLT and OSU, NRL would serve as a technical consultant to OSU. Representatives of OSU, NRL, and ONR then met at CINCLANTFLT in Norfolk, where CINCLANTFLT personnel described several existing applications which could be considered for the experiment, and OSU personnel further explained the properties of HSDMS with respect to various characteristics of the proposed applications. The representatives then formally determined the roles of their organizations and agreed on an application for testing. In the summer of 1975 two reports were prepared by NRL — one describing the Navy application to OSU, and another containing a proposed demonstration plan, including a database design, to be used on HSDMS. Meanwhile, OSU completed the implementation and testing of HSDMS, prepared database designs, and devised methods of accommodating various system transactions outlined in the demonstration plan. The demonstration took place on 7 January 1976 at NRL, in Washington, D.C.

#### A REAL-WORLD DATABASE APPLICATION

We felt that the database application chosen for this experiment should have several important characteristics:

- It should contain a reasonably complex data structure, which would serve to test the experimental system in supporting complex data-management functions;
- It should require much real-time human decision-making, which would serve to test the flexibility of the experimental system in processing the data structure dynamically and in interacting with the user effectively;
- It should have difficult access-control problems, which would serve to test the ability of the experimental system in supporting complex access-control requirements.

The chosen Navy application was ocean surveillance. Ocean surveillance involves maintaining location and status information on ships of all types and all nationalities (Fig. 1). Such information, crucial for wartime operations, is also useful in peacetime, for instance in search and rescue operations. The database supporting this application is implemented under the Honeywell Integrated Data Store (IDS) data-management system and will be referred to in this report as the Ocean Surveillance Data Base (OSDB). The entire OSDB structure contains over 100 record types and a similar number of IDS *chains* (in CODASYL terminology [7] these are *set types*).

#### Overall Access-Control Problems

Several serious access-control problems are associated with the operation of OSDB. In general, these problems stem from the inability of existing operating and database-management systems to discriminate reliably among data of different classifications and users of different clearances. Furthermore, these systems can neither reliably associate users with the data to which they have an access requirement nor reliably prevent users from accessing data for

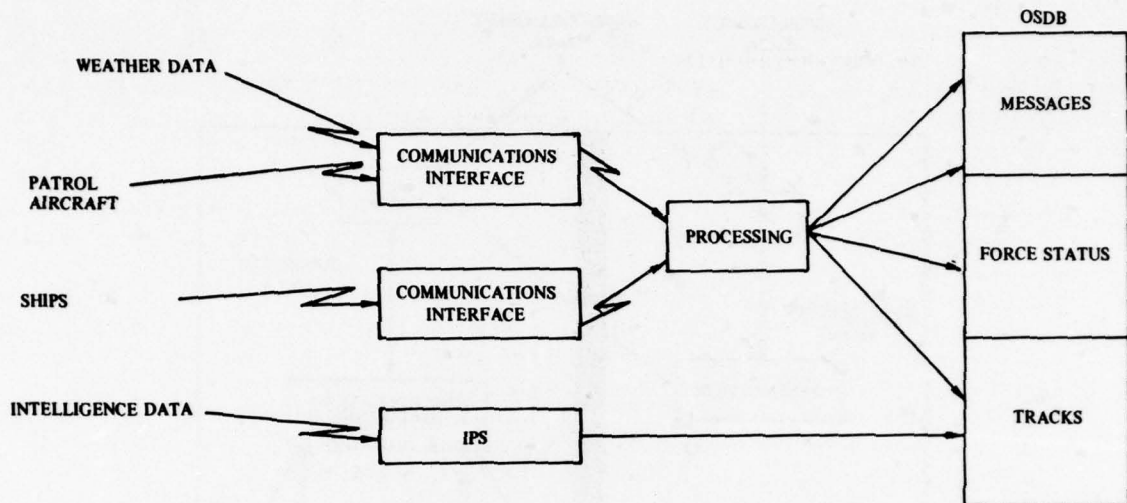


Fig. 1 - Data sources for the Ocean Surveillance Data Base (OSDB)

which they have no access authorization. In most cases, the solutions to these problems have been brute-force solutions, involving redundant hardware, software, and data and unnecessary data classification and user clearances. Let us focus our discussion on three specific problems and their brute-force solutions on the existing systems.

#### *Redundant Hardware, Software, and Data*

The first problem is that the access-control requirement for intelligence data is particularly strict. Both the identities and the capabilities of intelligence sources must be protected from discovery by unauthorized persons. It is therefore particularly important to be able to discriminate between intelligence and nonintelligence data and to correctly control access to this data. In the OSDB installation, separation of intelligence data and its users from nonintelligence data and its users is maintained by physical means. This situation is illustrated by Fig. 2, which shows some of the access-control-related features of the installation. Two separate systems exist. The Intelligence Processing System (IPS) processes both intelligence and nonintelligence data. OSDB processes nonintelligence data and "sanitized" intelligence data from IPS. (Data is "sanitized" by modifying it so that unique characteristics that reveal the source are removed.) Consequently, there is hardware and software redundancy and the need for a complicated sanitization interface between the two systems. Furthermore, there is data redundancy. Since both IPS and OSDB process nonintelligence data that the installation receives, large amounts of data in the OSDB database are also collected in the IPS database. When such data is updated, inconsistencies may arise unless extreme care is taken to update both databases in parallel.

Redundancy and inconsistency would be minimized if there were a single integrated database and a single database-management system that could discriminate reliably between intelligence and nonintelligence data, discriminate reliably among users with different clearances, and provide proper control over the users' access to the data.

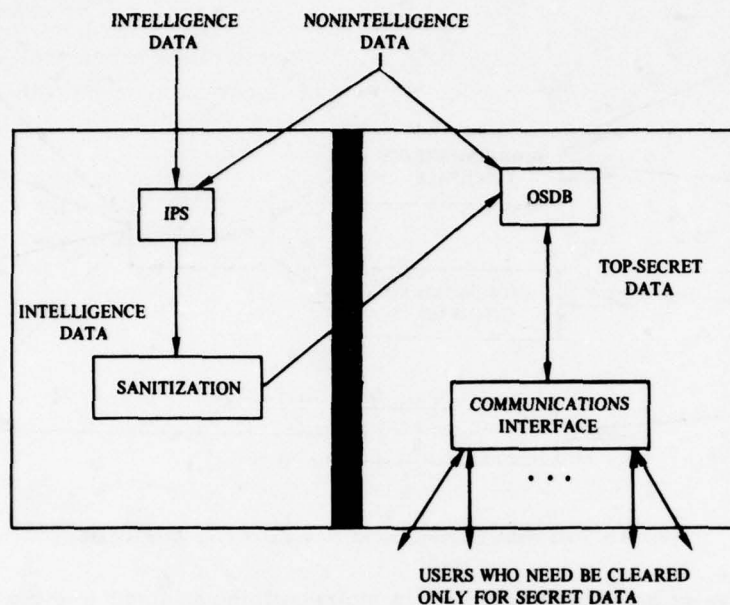


Fig. 2 - Partitioning of the OSDB installation for security

*Overclassification*

The second problem is prompted by the requirement, in processing classified data, to discriminate among data of different classifications and among users with different clearances. This requirement has an important impact on OSDB. The presence of top-secret data in the database raises the classification of the entire OSDB to top secret, even though most of the data is classified at a lower level. Consequently, terminals connected electrically to the database, and users of these terminals, have to be cleared to top secret, an expensive and time-consuming process. This problem is illustrated in Fig. 2: a group of users who are cleared to secret must access data which would normally be classified secret; however, because the data is held in OSDB, these users cannot access the data unless their clearances are raised to top secret.

*Lack of Multilevel Authorization and Control*

A third problem is that different authorities exert control over parts of the data within OSDB, since OSDB incorporates data from different sources and of different classifications. For example, separate authorities control the updating of the various parts of the database and it would be desirable for the existing system to enforce the updating constraints dictated by these authorities. Since the system is not able to do so, separate programs were written to update the various parts of the database and access to these programs is controlled externally. Retrieval programs, on the other hand, are given access to the entire database. The lack of multilevel authorization and control in the existing system makes it necessary to rely on redundant programming and manual control of program access to enforce the update constraints.

In summary, the brute-force solutions to the above problems were adopted because the present system provides no other solutions. A major purpose of the experiment was to show what could be accomplished with a system that could discriminate among data of different classifications and among users of different clearances, and could associate users with appropriate data under specific access-control requirements. *What we attempt to show is the functionality of a useful data-secure system. To achieve the required reliability, one must also show correct design and implementation.* This topic is discussed further in a later section (Comments on Demonstrations of Database Security Mechanisms). A discussion of the implementation and internal operation of the experimental system can be found in Ref. 2 and will not be repeated here.

### Concepts and Terminology

The OSDB is intended to support the on-line, interactive process of ocean surveillance. A major task in ocean surveillance is the association of contact reports of ships with ship tracks maintained in the database. The following terms and concepts are relevant to this process: A *contact report* is a single observation of a surface or subsurface ship under surveillance. Contact reports are sent to the OSDB in messages from the reporting sources. A *track* is a collection of contact reports about the same ship. The process of associating a contact report with a track in the database is called *contact correlation*. Correlation requires the comparison of contact-report data with the track data. The data used for such correlation include class, hull number, ship type, and flag for surface ships and label (which will be explained later) and sail number (the number on the conning tower) for subsurface ships. Furthermore, correlation considers whether the ship represented by a given track could have reached the position indicated in the contact report at the specified time of the contact.

Because positive identification of a surface vessel is likely, it is less difficult to associate a contact report of the surface vessel with the track representing its previous positions. This is hardly the case for subsurface contacts. An example of the correlation problem is shown in Fig. 3. With two ship tracks, 1 and 2, the problem is to determine the track to which a new contact belongs. The contact may represent a new position of the ship on track 1, a new position of the ship on track 2, or a new ship, not associated with either track. In the case of submarines, the correlation is more complicated. Since the data used for correlation is not necessarily unique to a particular submarine, the contact may potentially be in several tracks.

*Labels* are assigned to contacts by reporting stations and are included in contact reports by the reporting stations. Since many reporting stations may pick up the same ship, a given ship may have several labels. As an example of the use of these labels, assume that a ship leaves a port and is picked up by a reporting station. The reporting station may label the contact A and may continue to send out updated contact reports on A. As the ship moves on, it may also be picked up by another reporting station, which labels it B. The second station will continue to send out updated contact reports using the label B. Thus, this particular ship would have the labels A and B in its contact reports. Correlation must associate these individual labels into a track using position and other information provided in the contact reports. In this example, since A and B represent the same ship, they should be part of the same track. When a contact report with a new label (e.g., B) enters the OSDB, the analysts may immediately be able to correlate the report with an existing track (e.g., to associate B with A). If they cannot correlate a report, they must wait for additional reports containing data that may help

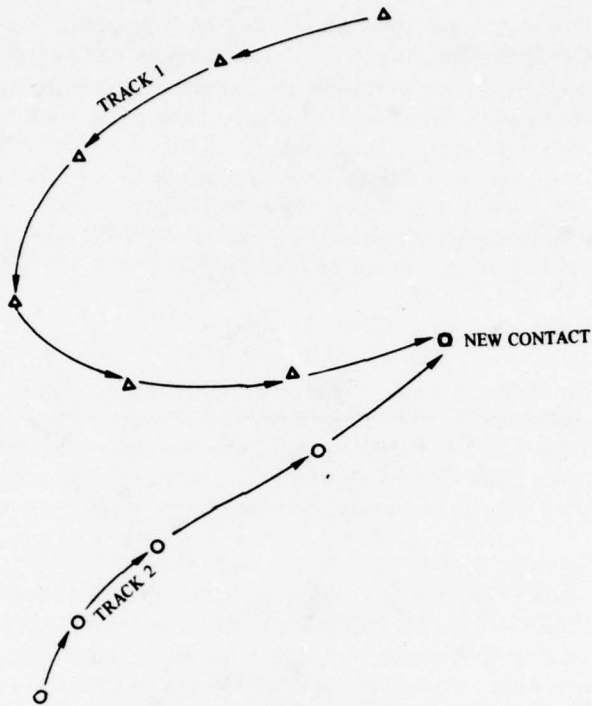


Fig. 3 - Correlation problem: Does the new contact belong to Track 1, Track 2, both or neither?

them to determine whether the first report was a continuation of an existing track or the beginning of a new track.

Whenever contact reports with different labels are correlated a *collective label* is assigned to the correlated reports. Collective labels may be used for a number of reasons. For example, a level of command may correlate data only from stations within its jurisdiction. It might then give its own collective labels to correlated data from those stations. Higher levels of command might then associate such collective labels into even higher level associations, with their own collective labels.

In the OSDB, there may be an arbitrary number of levels of collective labels. Each level is assigned a level number. Thus, a level-1 collective label is formed by associating labels used by individual reporting stations. A level-2 collective label is formed by associating level-1 collective labels, and so on. These labels then form a hierarchical structure.

#### Database-Management Capabilities Required for Contact Processing

Since the correlation of contacts with tracks in the database involves much human decision-making, the following database-management capabilities are required to assist the analysts:

- The capability to search and display candidate tracks based on various conditions. These may be candidate tracks having data matching that of the contact report, or candidate tracks whose current positions are within a given area surrounding the position of the contact report.
- The capability of adding new contact reports.
- The capability of associating one contact report with other reports in the same track, once an analyst has made a positive correlation.
- The capability of reassociation based on new data. The main reason the capability for reassociation is needed is that contact reports will not always enter the system in the same order as the date-time groups of the reports would indicate. This is because some of the reporting stations require longer analysis of their data than others before sending out a report. A report that arrives after the formation of a track may invalidate that track. Another reason is that surface contacts may be mistaken for subsurface contacts. Thus, a surface vessel may be tracked for some time as a submarine.

Thus, the database management system must assist the user by allowing him to examine relevant data and by implementing his decisions with respect to that data. Since the data is less positive than in many other applications, the data structure used to support this application must be very responsive to user control.

#### AN EXPERIMENTAL SOLUTION TO THE DATABASE APPLICATION

In this section, we describe the application as implemented on the HSDMS for this experiment. Differences between the application as implemented on HSDMS and the real application previously described are the following:

##### *Access Control*

As noted previously, the OSDB is only a part of the installation supporting the ocean-surveillance application. Further, the OSDB has a large, complex data structure, and contains many different kinds of data. There are specific access-control problems associated with the use of various small parts of the OSDB, but the more interesting access-control problems are those previously described which concern the entire installation, including both OSDB and IPS. In an effort to re-create these problems without including the entire OSDB database, we incorporated directly into the application additional access-control requirements, users, and data. This enabled the application as implemented on the HSDMS to provide a more interesting test of the HSDMS access-control capabilities. The experimental database was therefore made to contain some intelligence data, as well as force-status (FS) data which could be of either secret or top-secret classification. (In a slight simplification of the actual government security procedure for intelligence data, "intelligence" was considered as another security classification, higher than top secret. Thus, the experimental database consisted of data classified, for purposes of the experiment, as "secret," "top secret," or "intelligence." No actual classified data was used in the experiment.) Such a mixture of data would have to be supported by a system providing a true multilevel access-control capability. In addition, the various categories of users

(secret-level users, top-secret-level analysts, users cleared for intelligence data, etc.) were assumed to access the experimental database directly. Again, a system with a true multilevel access-control capability would have to ensure that users with proper clearances were given appropriate use of the data for which they were cleared.

#### *User Interface*

Instead of providing a tailored transaction-based interface as used in the system being emulated, the experimental system HSDMS provides a generalized query interface. This interface was not tailored to the ocean-surveillance application and is rather primitive. It was felt that, with a limited time available, effort on HSDMS should be concentrated on providing basic, but generalized, capabilities. More human-engineered or tailored interfaces could be easily implemented on HSDMS, using the basic capabilities provided.

#### *Data Structure*

The IDS system that supports the existing application uses named linkages (called *chains*) to link together associated records. An application program that stores a record must ensure that the record is also made a part of the appropriate chains. A chain can then be traversed by application programs to access all the records on the chain. A different approach is used in HSDMS to achieve the same effect. In HSDMS, records which contain the same attributes are automatically linked together if the values of the attributes are found to be identical. Thus, for example, if PROPULSION is an attribute, and if two records contain an identical value DIESEL for the PROPULSION (shown as PROPULSION=DIESEL), the two records will be automatically linked together by the system. Such attribute-value pairs are called keywords. Keywords are used to denote the properties of the record and the record type (e.g., whether the record is a message or contact report). Using this approach, query expressions may be formulated based on content. When these expressions are entered through the terminal or presented by the application programs, the system will automatically retrieve all associated records.

In the following sections, we provide more detailed descriptions of the application as it was implemented on HSDMS for this experiment.

#### **Basic Elements of the Experimental Solution**

Messages enter HSDMS from outside sources and are placed in a message file. Most messages are of two general types: contact reports, and updates to information in force-status records. Once in the message file, the messages are extracted by analysts and staff personnel for appropriate processing. Analysts extract contact reports from the messages and add them to a track file, where they are then associated by label and by higher level associations (various levels of collective labels as appropriate) which connect information received from multiple sources about the same ship. As new contact information comes in, these associations are adjusted, created, or broken up to reflect the current best estimates of the information about the various ships under surveillance. The analysts also process force-status messages and update force-status records in the track file on the basis of information in those messages. The analysts are cleared to the top-secret level.

Controlling the efforts of the analysts are staff personnel who approve the formation of higher level associations and verify the correctness of data input to HSDMS based on sources which are unavailable to the analysts (e.g., intelligence sources). Staff personnel are cleared for intelligence data. The track file is interrogated by various users who wish to know the positions and other data concerning contacts. These users are cleared to the secret level. Overseeing the system operation are administrative personnel who delegate access rights to the other types of users.

Subject to the detailed access constraints listed below, users, analysts, and the staff have read and write access to the message file. In addition, users have read-only access to the track file, and analysts and the staff have read and write access to the track file. Administrative personnel have no access to either the track or message file. The detailed constraints are as follows:

- Within the message file, some messages are for staff only, and messages may be of various security classifications.
- Force-status records are classified top secret. Other track records are classified secret, top secret, or intelligence. To support the distinctions in security classification and record type, each record will contain a field for the security classification and record type.
- Only the staff may update certain designated fields in records (e.g., staff-approval fields and security classifications).
- Analysts may not form associations with collective labels higher than level 1 unless they have been approved by the staff. Similarly, analysts may not delete or remove components from such associations without staff approval. Analysts, however, may create and delete level-1 collective associations without reference to the staff.
- Analysts may not delete records which are marked as staff-approved. This allows the staff to confirm a contact report that is verified by intelligence data or by other data not available directly to the analysts. Records which are not confirmed by the staff in this way may be deleted by the analyst in the exercise of his judgment, subject to other access constraints.
- Analysts may not assign labels to collective associations higher than level 1 but may only request permission to form them. The names are assigned by the staff.
- Users may not access data outside their security clearances or outside any other defined constraint.

### **The Database Organization**

The database organization for the application can be described in terms of field specifications, record types, and file structures. To avoid unnecessary detail, we will not show their storage layout.

*Records and Fields*

As depicted in Fig. 4, there are six major record types. The TY field holds the record type, and the SEC field holds the security classification of the record, for all types. A contact record is used to hold data about a single contact report. In a contact record, the LABEL field is a keyword field. An organization record is used to associate contact records of different labels into higher level associations. The organization record which denotes a level-1 association contains a LABEL field containing the collective label, a level field indicating the level of association, and other LABEL fields for each associated lower level label. A similar arrangement allows for the construction of higher level associations. The association record is used by analysts to request staff approval of certain types of associations. The data contained in the record depend on the requested action. Both the contact and organization records contain an APRV field for the staff to indicate approval or disapproval of the record. The APRV field in the association record allows the staff to indicate approval or disapproval of the request. The field may be written or altered only by the staff. However, the analyst may display the field. The records also contain a level field indicating the level of association desired.

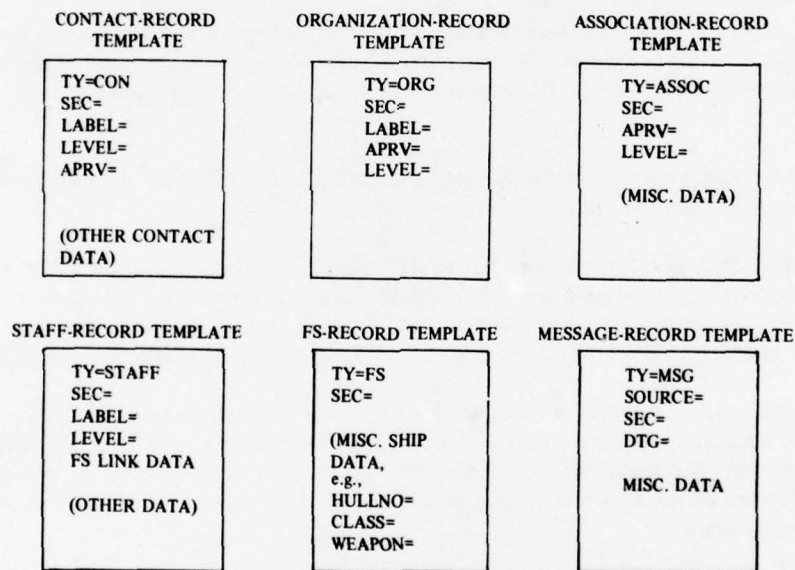


Fig. 4 - Record contents in the experimental database

The staff record is created when a specific ship or ship class can be associated with a collective label. It is assumed that if sufficient data is available to identify a particular ship or ship class with a track, then there is sufficient justification for assigning at least a level-1 label. The FS (force-status) record is used to hold data about the characteristics and status of individual ships. This record is associated with the staff record of a particular track. The message record is used to hold contact reports or force-status data for input to the database. Other types of data specific to the staff, and of various classifications, will be contained in the message records as well.

### File Structure

The file structure for the application allows the association of the aforementioned types of records for processing. We illustrate the file structure by means of an example showing the steps in the construction of a level-3 association.

In Fig. 5, contact records are entered into HSDMS and linked automatically by common label. All contact records have the keyword `TY=CON` in common, but, for clarity, links for this keyword are not shown. At this point, a query of the form `LABEL=A1` would retrieve the four records depicted in the first column of Fig. 5. Note that a noncollective label is denoted by `LEVEL=0`.

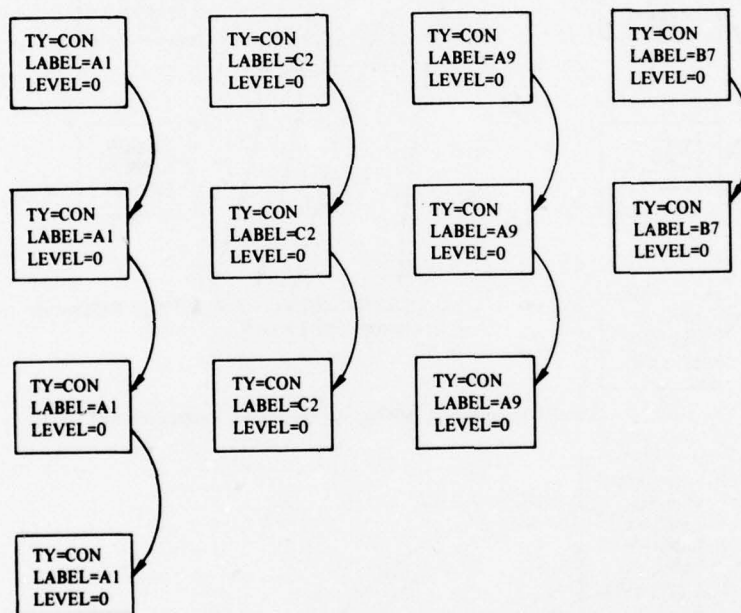


Fig. 5 - Contact records linked by labels

In Fig. 6, the records in Fig. 5 have been organized into two level-1 associations (X and Y). An organization record containing the level-1 label is created to represent each association. Included in each organization record are the labels which denote the records in Fig. 5 belonging to the association. Contact records which have been entered with the level-1 label are also shown in Fig. 6. At this point, a query of the form `LABEL=A1` would retrieve not only the records (in Fig. 5) reported with that label, but also the organization record for X. This would, in turn, allow the user to retrieve the contact records reported with label X and other labels participating in the same association.

In Fig. 7, the two associations in Fig. 6 have been associated to form a level-2 association. This is done by combining the contents of the two lower level organization records into the new organization record. The lower level organization records are then deleted. Contacts reported with the collective label `PAC7` are also shown in Fig. 7. At this point, a query for any

lower level label associated with PAC7 will also retrieve the PAC7 organization record. This allows the user accessing the data at one level to move up or down in the associational hierarchy to other existing levels, subject to access constraints.

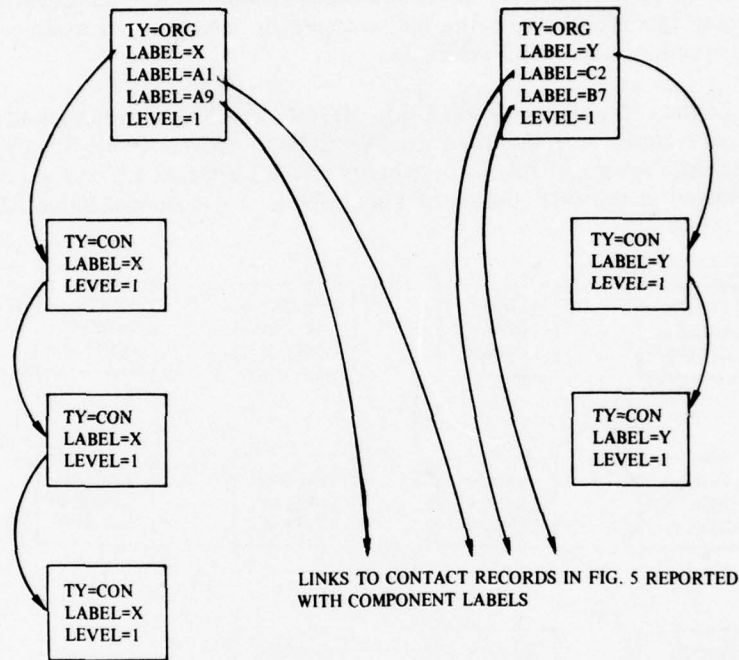


Fig. 6 - Level-1 records and linkages to associated contact records

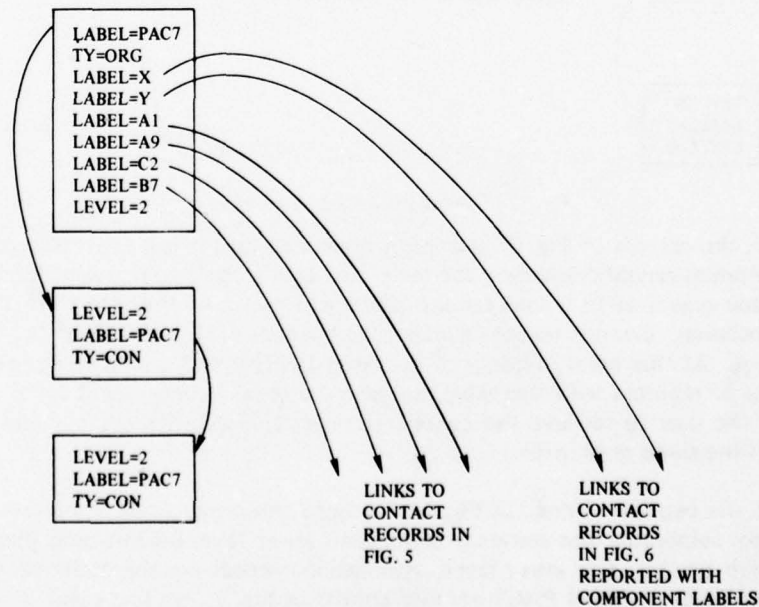


Fig. 7 - Level-2 records and linkages to associated contacts

In Fig. 8, the level-2 association has become part of a level-3 association SPOT-12. Again, the organization records at the previous level are combined and deleted. In this example, there was only one level-2 label. However, other lower level labels could be added by simply updating the organization record.

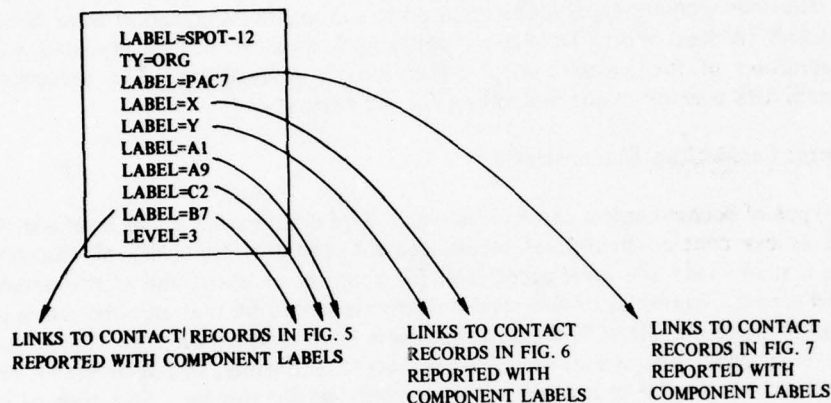


Fig. 8 - A level-3 record and linked to associated contacts (note: no contact records have been reported with the level-3 label)

#### *The Staff-Approval Process*

As noted earlier, analysts may not form associations higher than level 1 without staff approval. The process of obtaining staff approval adopted in HSDMS shows how such administrative rules can often be enforced in a database by simple access constraints. The process consists of the following steps:

**Step 1:** For his request, the analyst creates an association record (i.e., TY=ASSOC) which contains the fields (namely the LEVEL and LABEL fields) of the organization record he wishes to create. He is allowed neither to create nor to see the SEC and APRV fields. Furthermore, he is not allowed to assign any name to the association.

**Step 2:** The staff periodically retrieves all association records. Those association records with no APRV field represent outstanding requests from analysts. If a request is to be approved, a staff member places in the association record a LABEL field containing the name he has assigned to the organization, specifies the appropriate security classification, adds YES to the APRV field, and changes the TY field from ASSOC to ORG. Consequently, the requested organization record is created for the analyst. If the staff member disapproves the request, he merely adds APRV=NO to the association record.

**Step 3:** By retrieving all association records satisfying the query APRV=NO, the analyst learns which requests have not been approved by the staff. Organization records representing approved associations may be retrieved by the query APRV=YES. Since analysts may not update staff-approved records, the association approved by the staff cannot be changed by the analyst. Analysts are prevented from creating high-level associations directly by an access constraint that they may not create records with TY=ORG and LEVEL > 1.

## DEMONSTRATION OF THE SOLUTION

Given the database-design and access-control requirements described earlier, HSDMS had to be demonstrated to provide a number of capabilities to support the application. These capabilities are divided into two classes: data-management capabilities, and access-control capabilities. Basic data-management capabilities required to support the application were discussed in a previous section (A Real-World Database Application), and will not be repeated here. However, a description of the access-control capabilities is given here. The presence of such HSDMS capabilities was the major motivation for the experiment.

### Access-Control Capabilities Demonstrated

Four types of access-control capabilities were to be demonstrated, file-level access control, record-level access control, field-level access control, and the capability of changing access rights. The first of these, file-level access-control, protects an entire file of the database from unauthorized access. Examples of file-level constraints would be that administrative personnel may not read or write the TRACK file and that users have read-only access to the TRACK file. Under a file-level constraint, the entire file is protected uniformly, which precludes any condition being placed on the file to qualify specific records within the file. This type of protection is often seen in contemporary commercial systems.

Record-level access control protects various collections of records within a file from unauthorized access, and is a feature not often seen in contemporary commercial systems. It is required for the control of access to certain types of records (e.g., analysts may not create level-2 organization records) or for the control of access to records with specified classifications (e.g., certain users may not see records classified top secret). Access constraints at the record level may use any Boolean expression of field name (attribute) and value as a qualifying condition, provided that the field name has been declared to the system for that purpose. Once a record has been qualified by a particular record-level access constraint, the entire content of the record is protected uniformly as defined in the constraint. For example, if the constraint is that a certain user may only read contact records, then the user will not be able to write any field of a contact record.

Field-level access control protects specific fields within records from unauthorized access. Examples of field-level constraints are that analysts may not write into staff-approval fields and only staff may write or change security classifications. Naturally, field-level security applies only to records which the user is permitted to access on the basis of his record-level security constraints.

To support real-time command and control, HSDMS provides the on-line capability to change access rights while the system is operating. This eliminates the need to bring the system down, to change access rights, and to restart the system. A number of advantages are associated with this capability. A user may find that he is unauthorized to perform some operation which he has the need to perform. He may immediately request and receive the right to perform the operation from the appropriate authority for as long as he needs it. As soon as there is no such need, the right to perform the operation may be withdrawn. New users may be granted access to the system immediately when the need arises, rather than having to wait until some time convenient to the system. By careful assignment of access rights, users may

be guided through certain administrative chains of approval by the system before being able to perform certain database accesses. Thus, the system may assist in the enforcement of certain administrative rules.

### **Demonstration Plan**

All steps in this plan were successfully carried out during the demonstration of HSDMS on 7 January 1976. A brief narrative description of the steps in the demonstration plan is given below. The appendix contains sets of HSDMS queries corresponding to some of the steps in the demonstration and illustrates how these steps were accomplished using HSDMS.

Step 1: An analyst retrieves messages containing contact reports (CRs) from the message file. He then manually extracts the CRs. He adds the new CRs he extracted from the messages to the track file. He then updates the messages to indicate that CRs have been processed. The relevant labels are then displayed showing that proper record linkages have been established by the system.

Step 2: Two labels from different sources are displayed and formed into a level-1 association by an analyst. The level-1 organization record and all associated records are then displayed to show the results of the association.

Step 3: An analyst attempts to update staff-approval fields in a CON record and is rejected by the system.

Step 4: An analyst displays two level-1 associations and attempts to create a level-2 association without staff approval, and is rejected. He then issues a request by creating an association at level 2.

Step 5: An analyst retrieves a label with intelligence CRs associated with it. He receives only the nonintelligence part of CRs. The staff retrieves the same label and receives all the data in the CRs, including the intelligence data.

Step 6: The staff updates some staff-approval fields that the analyst tried to update in step 3. The update is successful.

Step 7: The staff retrieves records waiting for staff approval, including the request created by the analyst in step 4. The staff rejects one association and approves the one created in step 4.

Step 8: The staff retrieves all intelligence organization records. The staff then associates two into a level-2 association, without going through the staff-approval process required for analysts. The association is successfully created.

Step 9: A user tries to retrieve an FS message and is rejected.

Step 10: An analyst, a user, and the staff all display level-3 organization records. The staff receives more of them, since some are intelligence data. The staff raises the classification

of one of the level-3 associations to intelligence data. Both the analyst and the user redisplay the organization records. Neither now sees that specific association.

Step 11: Administrative personnel change the access right of a user to include rights to intelligence data, but without authority to see FS records.

Step 12: The user and an analyst retrieve all level-3 associations. The user should get associations which the analyst does not get, namely, associations classified as intelligence data.

Step 13: The user tries to access FS records, and is rejected, even though he has a higher clearance than the classification of FS records.

Step 14: Administrative personnel change the rights of the user back to secret clearance.

Step 15: The user tries step 12 again. This time he does not receive associations classified as intelligence data.

Numerous record retrievals by content were demonstrated, such as records retrieved by their type (e.g., messages containing contact data) and by type and label (e.g., contact records pertaining to a particular label). All of this retrieval was confined within the user's access-control constraints. The effect of these access-control constraints was demonstrated by having two users with different constraints retrieve the same data (i.e., issue the same request) and comparing the resulting output (e.g., in step 5).

Early in the demonstration, new contact records, corresponding to new contact reports, were stored in the database (in step 1). These records were linked automatically by the system to other records in the database having the same label. This was shown by performing a retrieval on the label following the storage of the new records. The ability of the system to accept new contact data and to associate this data automatically with other related data is a basic requirement in the application.

Another basic requirement in the application is the ability of the system to support association of contact data into higher level associations, as directed by users. Several such associations were performed during the demonstration. One type of association, the association of labels into level-1 associations, was performed by an analyst alone (in step 2). In this case the system allowed the proper organization record to be created and automatically linked the new organization record and the existing contact records to be associated into the association. The existence of the linkages was shown by the appropriate retrieval request. A second type of association is association of level-1 associations into those of higher levels. Since association at these levels must be approved by the staff, the analyst could not create the appropriate organization record directly. During the demonstration, the analyst attempted to form a level-2 organization record, but the request was rejected (in step 4). However, it was then shown that when the analyst followed the proper procedure for interacting with the staff, the associations were created (in step 7).

Finally, some of the users had their access rights changed during the demonstration. That this process worked was shown by having those users issue the same requests before and after the change (in steps 11 through 15).

## RESULTS OF THE EXPERIMENT

The direct result of the experiment was the successful demonstration of the HSDMS security facilities for the selected application. We consider the experiment a successful illustration, for a realistic application, of the principle that a single database may be made to appear differently to different users based on security considerations. This was accomplished through the use of a flexible access-control mechanism, allowing the declaration and enforcement of security constraints based on the logical structure of the database. With such a capability, there is no need to have physically different databases for different classes of users, based on security requirements. Such data (and often hardware) redundancy characterizes many existing systems. At the same time, however, the successful demonstration reported here is not to be considered a successful test of the "security" of HSDMS. As we noted in a previous section (Overall Access-Control Problems), correct design and implementation are also important requirements for a useful data-secure system. The next section (Comments on Demonstrations of Database Security Mechanisms) expands on this point to place in perspective demonstrations such as the one described in this report.

In addition to the primary results of the experiment, the experiment also had the effect of confirming our views on a number of related issues. These issues will be described in the following subsections.

### Data-Structure Transformation

As already described, part of the experiment consisted of converting an application from a DBMS using explicit named interrecord relationships (specifically, IDS chains) to a DBMS using implicit interrecord relationships based on shared values. Intuition says that this conversion can always be performed, and often in a fairly straightforward way. Intuition proved to be correct in this experiment. It became very clear that value-based structures could always be developed to support the application requirements, and that the transformation could be made either in a straightforward way, preserving all the structure contained in the original explicit relationships, or in a more or less complex way. No formal or automated approach to performing the conversion was used in the experiment. Instead, the original structure was analyzed to determine its properties, and then value-based structures were developed to reflect those properties of the original structure which were necessary for the experiment. (Not all the capabilities of the original structure were required.)

### Relationship of Security and Integrity Constraints

The experiment had the effect of reinforcing our opinion that security and integrity constraints are inherently similar and that the same mechanism should deal with both. This was illustrated in the experiment by the use of "security" constraints to enforce what was essentially an administrative procedure, namely, the staff-approval requirement for formation of certain types of associations. Similar types of constraints could be used to enforce message review and approval procedures in message-handling systems, or other types of integrity-preserving procedures. The close relationship between security and integrity constraints has also been dealt with elsewhere in the literature [8,9].

### Usefulness of Value-Based Access Control

The experiment also had the effect of reinforcing our opinion that it is very useful to be able to express security constraints using the same kinds of conditions used in queries and other data-manipulation operations. This facility helps in some sense to insure that the access-control mechanism is equal in power to the data-access mechanism, since both have the same ability to qualify records. It also allows the same parsing mechanism to be used for both access and access control, allows the use of such techniques as "query enhancement" [9] in enforcing the constraints, and reduces the number of languages that must be learned to use the system. The latter is a particularly important point when the system may be used for "personal" databases, or databases that have individual owners, rather than being under the control of a system-wide data administrator. It is similarly important when many different people may have security responsibilities for different parts of a database. In such cases, it is important that individual users be able to use the access-control mechanism, just as they do the data-access mechanism. The assumption that individual users will control other users' accesses to databases they own is a fundamental one in System R [8].

### COMMENTS ON DEMONSTRATIONS OF DATABASE SECURITY MECHANISMS

As we noted previously all the steps in the demonstration plan were successfully carried out using HSDMS during the course of the demonstration. However, it was not to be expected that such a demonstration, exhibiting as it does only the more "external" characteristics of a system, would be in any sense a demonstration of the security of the system. The ability of data-security facilities such as those in HSDMS to be used in practice to enforce security constraints depends upon the ability to produce systems which can be relied on to enforce the declared constraints, and upon the ability to protect the environment in which these systems must operate. The following paragraphs amplify this point in order to place in perspective demonstrations such as the one described in this report.

#### Factors in Providing a Secure Database Facility

Many factors go into providing a secure database facility besides the existence of a DBMS with an access-control mechanism which appears to provide the necessary functions. For example, there must be protection of the environment in which the system is to operate. Appropriate levels of physical protection of the database, hardware, software, and communication lines from tampering or penetration must be provided. The hardware on which the system is to run must be reliable and must provide necessary protection mechanisms. In addition, both the operating system and the DBMS must be reliable in their operation and must be free from subtle design flaws, either external or internal, which would allow unauthorized access to data. For example, the operating system must protect the DBMS from concurrently running programs and must insure that application programs which access the database can do so only through the DBMS. The DBMS itself must not only provide the necessary facilities, but it must provide them reliably and in a manner which cannot be circumvented. Moreover, these various components must not only exist, but they must be shown somehow to provide the appropriate levels of protection.

Some of these components can be proven adequate by testing. For example, testing can be used to show that hardware or communications lines do not radiate electronically in such a

way as to allow a penetrator to listen in on the system's operation. However, even though software testing is certainly important and useful, it has proven inadequate in the past as an approach for showing that a piece of software, such as an operating system or DBMS, will provide proper security (see Ref. 13). Since a demonstration is in essence a test, it is similarly true that a demonstration cannot be relied on to show that a system is secure. Because of the problems of showing security by testing, the "security kernel/program verification" approach has gained influence in the design of secure operating systems, and, because the problems of secure database-management systems are similar, this approach must be considered in the DBMS context as well.

### **The Security-Kernel Approach**

In the security-kernel approach, an operating system is designed so that the security-relevant portions of the code are condensed into a security kernel which is small enough to be proven correct by formal program-verification techniques. Once the security kernel is proven correct, it is believed that there should be confidence that the kernel will always work properly. Thus, the rest of the operating system, the part not proven correct, may still have undetected flaws, even though presumably subjected to modern software design and test techniques. But these flaws would not affect the security of the system, since all the security-relevant code would be contained in the kernel, would function properly, and would prevent any breach of security that might otherwise result from them.

It has been suggested that this same approach can be applied to DBMS security. For example, it has been suggested that the security kernel of a secure operating system be used for all security checking, including that of the DBMS, thus requiring no security-relevant code in the DBMS itself [14]. This is seen as desirable because there would be no need, as far as security is concerned, to prove the correctness of any part of the DBMS as long as the operating system security kernel has been proven correct. The approach described in Ref. 14 was designed to handle "military" security, i.e., classifications and categories. Others have suggested that the operating-system security kernel may not be capable of providing the more general types of database access control, e.g., the general types of data-dependent access control provided in such systems as HSDMS, INGRES [9], and System R [8], as opposed to the strict military classification scheme. Among these people, there are two points of view. One is that it may be possible to develop a DBMS security kernel, similar in function to the operating-system security kernel, but adapted toward DBMS protection requirements. The other point of view is that this kernel, which would collect all the security-relevant parts of the DBMS, would prove to contain a large percentage of the DBMS code in order to allow data-dependent security checking. Thus, such a kernel would not really be a kernel at all, as it would contain more code than could be proven correct using current verification techniques. These people think instead that the system will simply have to be assumed trustworthy, although again it would be subjected to modern software design and test methods.

A compromise position, which we feel is useful, is that even though it might not be possible to reduce the amount of security-relevant code to the point where it would constitute a kernel and be provable using current verification techniques, it might still be possible, by proper design, to reduce the amount of security-relevant code to some extent. A possible approach might involve tradeoffs between the algorithms implemented in the DBMS software and the

data structures to be manipulated by that software, as is suggested in Ref. 5. A DBMS containing a reduced amount of security-relevant code might be provable given future improvements in program verification techniques, even though it might not be provable today. Of course, it remains to be seen whether such approaches can be successfully applied to general-purpose database-management systems.

### External System Characteristics

The above discussion has concentrated on the internal characteristics of the system. We are convinced that the external characteristics of the system also play a role in determining the system's security and in determining how difficult the testing of security is. For example, we are convinced that a properly designed query-language (or other high-level) interface provides enhanced security over a lower level interface, provided that the implementations are equally reliable. This is because fewer of the system's internal operations are exposed at a higher level interface and it is easier to control precisely what a system user can do at the interface [15]. Such interfaces may also be more easily evaluated by proof techniques aimed at ensuring that the interface has no "holes" or design flaws that can be exploited by system penetrators. However, it is not yet clear what a properly designed interface is in this context. It is also not necessarily true that any high-level interface is by definition foolproof. This assumption was once made with respect to programming languages (namely, that a higher level programming language automatically provided more security). However, studies (such as Ref. 11) have shown that such languages sometimes have design flaws or features that allow the interface provided by the language to be penetrated, allowing machine-language programming to take place. Attempts to penetrate DBMS systems providing high-level interfaces might be useful. However, current results do indicate that such systems in fact provide greater security, assuming attack only through the DBMS itself.

### Effect of These Factors on the Experiment

From the preceding discussion, it should be clear why the demonstration reported on here cannot be considered a satisfactory demonstration of the security of HSDMS. For example, the demonstration was really only a single test of the system. A number of such tests would be necessary. In addition to such tests, the interface languages of HSDMS would have to be subjected to intensive scrutiny to ensure that the interfaces were all completely defined in an unambiguous way and that contradictory declarations of access constraints could not cause abnormal system operation. The implementation of HSDMS would also have to be subjected to intensive scrutiny to ensure reliable operation and correct implementation. It might be necessary to reorganize the system so as to concentrate security-relevant code for purposes of program verification. The assumptions which HSDMS makes about the functions to be performed by the operating system would also have to be completely and precisely specified.

It should also be clear from the preceding discussion that even a satisfactory proof that HSDMS was secure would not necessarily mean that HSDMS could replace the existing data-management system in the Navy application described here, and provide a secure database system. The security problems in the existing application are due to a great extent to the lack of a secure operating system, as well as the lack of a secure DBMS. A secure DBMS running under an insecure operating system would still not provide adequate security, since a penetrator could possibly exploit "holes" in the operating system to access the database directly, bypassing any

security constraints enforced by the DBMS. A DBMS and an operating system must cooperate in enforcing system security constraints, and both must be satisfactory from a security standpoint.

## CONCLUDING REMARKS

As we stated at the beginning of this report, our primary goal in the experiment described here was to show the applicability of advanced security features of an experimental DBMS to a real-world problem. We believe that the experiment has shown that such features, if properly implemented and verified, can clearly provide important benefits. These benefits include:

- elimination of redundant hardware, software, and data;
- automated enforcement of administrative controls;
- freer access to systems by wider classes of users with different access requirements and different security constraints; and
- elimination of unnecessary security-clearance costs due to overclassification.

Systems other than HSDMS have been and are being developed with similar security capabilities [8-10,12]. The benefits demonstrated here for HSDMS may be expected from these other systems for applications with complex data-security requirements, provided that corresponding requirements for reliability and verification can be met.

## ACKNOWLEDGMENTS

We acknowledge with thanks the efforts of the following individuals and organizations whose cooperation was essential to the success of the experiment described herein: Mr. Neal Kaffen, Ms. Anne Breene, and Mr. Don Schmaltz of the Ohio State University; Mr. Stanley Wilson and Dr. Joseph Kullback of the Naval Research Laboratory; Dr. Paul Patent of the Office of Naval Research; Mr. H. O. Lubbes of the Naval Electronic Systems Command; the Information Systems Office of the Commander-in-Chief, Atlantic Fleet; and the Navy Regional Data Automation Center, Washington, D.C. This work was supported in part by the Office of Naval Research under Contract N00014-75-C-0573 and was initiated by Mr. Marvin Denicoff, Information Systems Program, Office of Naval Research.

## REFERENCES

1. D.K. Hsiao and R.I. Baum, "Information Secure Systems," in *Advances in Computers*, Volume 14, Academic Press, New York, 1976, pp. 231-272.
2. D. K. Hsiao, "A Software Engineering Experiment in the Management, Design and Implementation of a Data Secure System," in *Proceedings of the 2nd International Conference on Software Engineering*, San Francisco, Oct. 1976, pp. 532-538.
3. H.R. Hartson and D.K. Hsiao, "A Semantic Model for Database Production Languages," in *Proceedings of the 2nd International Conference on Very Large Data Bases*, Brussels, Sept. 1976, pp. 27-42.

4. H.R. Hartson and D.K. Hsiao, "Full Specifications in the Semantic Model for Database Protection Languages," in *Proceedings of the 1976 ACM National Conference*, Houston, Nov. 1976, pp. 90-95.
5. E.J. McCauley, III, "Highly Secure Attribute-Based File Organization," in *Proceedings of the Second USA-Japan Computer Conference*, Tokyo, Aug. 1975, pp. 497-501.
6. R. I. Baum and D. K. Hsiao, "Database Computers — A Step Towards Data Utilities," *IEEE Trans. Computers*, C-25, 1254-1259 (1976).
7. CODASYL Data Description Language: *Journal of Development*, June 1973, National Bureau of Standards Handbook 113, U.S. Government Printing Office, Washington, D.C., Jan. 1974 (Available from the Superintendent of Documents, Washington, D.C., SD Catalog No. C13.6/2:113).
8. M. M. Astrahan, et al., "System R: Relational Approach to Database Management," *ACM Trans. on Database Systems*, 1, 97-137 (1976).
9. M. Stonebraker, "Implementation of Integrity Constraints and Views by Query Modification," in *Proceedings of the 1975 SIGMOD International Conference on Management of Data*, San Jose, Calif., May 1975, pp. 65-78.
10. F. Manola, "An Extended Data Management Facility for a General-Purpose Time-Sharing System," Report 71-24, Moore School of Electrical Engineering, University of Pennsylvania, Philadelphia, May 1971 (NTIS No. AD-724 801).
11. R. W. Conway, W. L. Maxwell, and H. L. Morgan, "On the Implementation of Security Measures in Information Systems," *Commun. ACM* 15, 211-220, (1972).
12. D. K. Hsiao, "Access Control in an On-Line File System," in *File Organization - Selected Papers from FILE68, An IAG Conference*, Studentlitteratur Ab, Lund, Sweden, Nov. 1968, pp. 246-257.
13. J. P. Anderson, "Computer Security Technology Planning Study," ESD-TR-73-51, Vols. 1 and 2, Electronic Systems Division, Air Force Systems Command, Hanscom Air Force Base, Bedford, Massachusetts, Oct. 1972.
14. T. H. Hinke and M. Schaefer, "Secure Data Management System," RAD-TR-75-266, Rome Air Development Center, Griffiss Air Force Base, New York, Nov. 1975.
15. F. Manola and S. Wilson, "Data Security Implications of an Extended Subschema Concept," in *Proceedings of the Second USA-Japan Computer Conference*, Tokyo, Japan, Aug. 1975, pp. 481-487 (also available as NRL Report 7905, July 1975).

## Appendix

### EXAMPLES OF HSDMS USAGE CORRESPONDING TO VARIOUS STEPS IN THE DEMONSTRATION PLAN

**Step 2—Two labels from different sources are displayed and formed into a level-1 association by an analyst. The level-1 organization record and all associated records are then displayed to show the results of the association.**

#### \*QUERY

REQUEST NAME : S02C1  
ACCESS TYPE : R  
FILE NAME : TRACK  
USER TEMPLATE : CNUSE

In this example, the user (an analyst) responds to the system's request for a new command (indicated by an asterisk) by indicating that he wishes to input a query (user responses are underlined). Following the command QUERY,

QUERY : LABEL=F3/LABEL=P4\*

\*STATUS

REQUEST NAME : S02C1

RECORD COUNT : 8

\*EXAMINE

REQUEST NAME : S02C1

Number of records to be displayed  
(CR implies all):

Increment (e.g., every nth record;  
CR implies all):

SEC : TSCRT  
TY : CON  
LONG : 429  
LAT : 292  
DTG : 15-Nov-7510:26  
SPEED : 26  
SENSOR : RADAR  
BEARING : 37  
LABEL : F3

SEC : TSCRT  
TY : CON  
LONG : 229  
LAT : 172  
DTG : 15-Nov-7510-26  
LABEL : F3

SEC : SECR  
TY : CON  
LONG : 394  
LAT : 94  
DTG : 15-Nov-7510:26  
LABEL : F3

SEC : SECR  
TY : CON  
LONG : 355  
LAT : 238  
DTG : 15-Nov-7510:26  
LABEL : F3

SEC : TSCRT

the system prompts the user for the necessary information to complete the request. By naming the request, the user can refer to the request, or the output of the request, in subsequent commands. Access type "R" denotes "read". The user template is an internal record kept by the system of which fields in various records the user is allowed to access. The template name "CNUSE" is verified by the system against the user's identification to ensure that the user is authorized to use that template. The query expression is given as a disjunctive normal form expression of keywords (here, "/" means "or").

The user then asks for the status of his request, referring to the request name. If the request is complete, the system responds with the number of records retrieved. The user then asks to examine (have the system print out) the retrieved records, and indicates how many, and which, records are to be printed. The records are then printed out on the terminal.

MANOLA AND HSIAO

TY : CON  
LONG : 445  
LAT : 94  
DTG : 15-Nov-7510:26  
SPEED : 10  
SENSOR : RADAR  
BEARING : 264  
LABEL : P4  
HULL NO : 415

SEC : TSCRT  
TY : CON  
LONG : 314  
LAT : 257  
DTG : 15-Nov-7510:26  
LABEL : P4

SEC : SECR  
TY : CON  
LONG : 281  
LAT : 133  
DTG : 15-Nov-7510:26  
SPEED : 29  
SENSOR : RADAR  
BEARING : 211  
LABEL : P4  
HULL NO : 415

SEC : SECR  
TY : CON  
LONG : 456  
LAT : 187  
DTG : 15 Nov-7510:26  
LABEL : P4

Total number of records: 8  
Delete the retrieved record file: Y

\*ADD

REQUEST NAME: S02C2  
ACCESS TYPE: AS  
FILE NAME: TRACK  
USER TEMPLATE: CNORG

Enter attribute names and values  
as prompted.  
To signal completion of a record,

The records retrieved in response to a user's request are placed on a temporary file, which may be saved by the user or eliminated after the records have been displayed. In this case, the user indicates that he wants the temporary file containing these records deleted (the records still remain in the data base).

Here, the user creates the organization record which will establish the association between labels F3 and P4. He indicates his intent to add records by the command ADD, and the system prompts him as shown.

type a carriage return in response to the attribute name request.

ATTRIBUTE: SEC  
 TYPE(K,S,carriage return): S  
 VALUE: TSCRT  
 ATTRIBUTE: TY  
 TYPE(K,S,carriage return): S  
 VALUE: ORG  
 ATTRIBUTE: LABEL  
 TYPE(K,S,carriage return): K  
 VALUE: F3  
 ATTRIBUTE: LABEL  
 TYPE(K,S,carriage return): K  
 VALUE: P4  
 ATTRIBUTE: LABEL  
 TYPE(K,S,carriage return): K  
 VALUE: G7  
 ATTRIBUTE: LEVEL  
 TYPE(K,S,carriage return): S  
 VALUE: 1  
 ATTRIBUTE:

SEC : TSCRT  
 TY : ORG  
 LABEL : F3  
 LABEL : P4  
 LABEL : G7  
 LEVEL : 1

Valid as entered: Y  
 Continue? N  
 Total number of records created: 1

\*STATUS  
 REQUEST NAME: S02C2

RECORD COUNT: 1

\*QUERY

REQUEST NAME: S02C1

\*STATUS  
 REQUEST NAME: S02C1

RECORD COUNT: 9

Here, the user inputs a sequence of attribute-value pairs to form the record content. In each case, he designates whether the attribute-value pair is a keyword (K), a security keyword (S, i.e., a keyword which will be used in an access-control constraint), or ordinary data.

Here the user has responded to the attribute-name request with a carriage return, and the system prints the newly created record.

The response "N" to "Continue?" indicates that the user does not wish to create any more records.

**\*EXAMINE**

REQUEST NAME: S02C1

Number of records to be displayed  
(CR implies all):

Increment (e.g., every nth record;  
CR implies all):

SEC : TSCRT  
TY : CON  
LONG : 429  
LAT : 292  
DTG : 15-Nov-7510:26  
SPEED : 26  
SENSOR : RADAR  
BEARING : 37  
LABEL : F3

SEC : TSCRT  
TY : CON  
LONG : 229  
LAT : 172  
DTG : 15-Nov-7510:26  
LABEL : F3

SEC : SECRET  
TY : CON  
LONG : 394  
LAT : 94  
DTG : 15-Nov-7510:26  
LABEL : F3

SEC : SECRET  
TY : CON  
LONG : 355  
LAT : 238  
DTG : 15-Nov-7510:26  
LABEL : F3

SEC : TSCRT  
TY : CON  
LONG : 445  
LAT : 94  
DTG : 15-Nov-7510:26  
SPEED : 10  
SENSOR : RADAR  
BEARING : 264

In order to check that the proper linkages were established between the newly created record and the ones previously retrieved, the same query is re-input. Since information for request "S02C1" has already been input, there is no need to reenter it. In this case, nine records are retrieved, instead of the original eight, and the records are displayed, showing that now the newly created record is also retrieved on the basis of the query.

LABEL : P4  
HULL NO : 415

SEC : TSCRT  
TY : CON  
LONG : 314  
LAT : 257  
DTG : 15-Nov-7510:26  
LABEL : P4

SEC : SECR  
TY : CON  
LONG : 281  
LAT : 133  
DTG : 15-Nov-7510:26  
SPEED : 29  
SENSOR : RADAR  
BEARING : 211  
LABEL : P4  
HULL NO : 415

SEC : SECR  
TY : CON  
LONG : 456  
LAT : 187  
DTG : 15-Nov-7510:26  
LABEL : P4

SEC : TSCRT  
TY : ORG  
LABEL : F3  
LABEL : P4  
LABEL : G7  
LEVEL : 1

Total number of records: 9  
Delete the retrieved record file: Y

**Step 3—An analyst attempts to update staff-approval fields in a CON record and is rejected by the system.**

**\*UPDATE**

REQUEST NAME: S03C1  
ACCESS TYPE: US  
FILE NAME: TRACK  
USER TEMPLATE: STUSE  
QUERY: LABEL = C4\*

Here the analyst attempts to update a record satisfying LABEL = C4. To update staff-approval fields, those fields must be in the user template he uses for the request. However, no analyst has a template with staff approval fields

**\*STATUS**  
REQUEST NAME: S03C1  
RECORD COUNT: 0

in it. Here, he has tried to use a staff user's template. The system has detected that he is not authorized to use that template and has rejected the request (by indicating a record count of zero).

**Step 5—An analyst retrieves a label with intelligence contact reports (CRs) associated with it. He receives only the nonintelligence CRs. The staff retrieves the same label and receives all the CRs, including the intelligence data.**

**\*QUERY**

REQUEST NAME: S05C1  
ACCESS TYPE: R  
FILE NAME: TRACK  
USER TEMPLATE: CNUSE  
QUERY: LABEL=B1\*

Here, the analyst retrieves all records satisfying LABEL=B1 and displays them. Note that five records were retrieved.

**\*STATUS**  
REQUEST NAME: S05C1

RECORD COUNT: 5

**\*EXAMINE**  
REQUEST NAME: S05C1

Number of records to be displayed  
(CR implies all):  
Increment (e.g., every nth record;  
CR implies all):

SEC : TSCRT  
TY : CON  
LONG : 67  
LAT : 9  
DTG : 15-Nov-7510:26  
SPEED : 28  
SENSOR : RADAR  
BEARING : 220  
LABEL : B1

NRL REPORT 8176

SEC : TSCRT  
TY : CON  
LONG : 62  
LAT : 5  
DTG : 15-Nov-7510:26  
LABEL : B1

SEC : SECRT  
TY : CON  
LONG : 11  
LAT : 46  
DTG : 15-Nov-7510:26  
SPEED : 12  
SENSOR : RADAR  
BEARING : 216  
LABEL : B1

SEC : SECRT  
TY : CON  
LONG : 24  
LAT : 82  
DTG : 15-Nov-7510:26  
LABEL : B1

SEC : TSCRT  
TY : ORG  
LABEL : I5  
LABEL : H5  
LABEL : I3  
LABEL : B1  
LABEL : D5  
LABEL : C2  
LABEL : C4  
LABEL : E5  
LABEL : B2  
LABEL : G2  
LABEL : D2  
LABEL : A2  
LABEL : FRA5  
LABEL : DUD1  
LABEL : SUP3  
LEVEL : 3

Total number of records: 5  
Delete the retrieved record file: Y

**\*QUERY**

REQUEST NAME: S05C2  
ACCESS TYPE: R  
FILE NAME: TRACK  
USER TEMPLATE: CNUSE  
QUERY: LABEL = B1\*

Here, a staff user retrieves all records satisfying LABEL = B1 and displays them. Note that seven records were retrieved and that the two records retrieved here which were not retrieved by the analyst for the same request are intelligence records.

**\*STATUS**

REQUEST NAME: S05C2

RECORD COUNT: 7

**\*EXAMINE**

REQUEST NAME: S05C2

Number of records to be displayed  
(CR implies all):  
Increment (e.g., every nth record;  
CR implies all):

SEC : TSCRT  
TY : CON  
LONG : 67  
LAT : 9  
DTG : 15-Nov-7510:26  
SPEED : 28  
SENSOR : RADAR  
BEARING : 220  
LABEL : B1

SEC : TSCRT  
TY : CON  
LONG : 62  
LAT : 5  
DTG : 15-Nov-7510:26  
LABEL : B1

SEC : INTEL  
TY : CON  
LONG : 28  
LAT : 57  
DTG : 15-Nov-7510:26  
LABEL : B1

SEC : INTEL  
TY : CON  
LONG : 1

NRL REPORT 8176

LAT : 52  
DTG : 15-Nov-7510:25  
SPEED : 14  
SENSOR : VISUAL  
BEARING : 223  
LABEL : B1

SEC : SECR  
TY : CON  
LONG : 11  
LAT : 46  
DTG : 15-Nov-7510:26  
SPEED : 12  
SENSOR : RADAR  
BEARING : 216  
LABEL : B1

SEC : SECR  
TY : CON  
LONG : 24  
LAT : 82  
DTG : 15-Nov-7510:26  
LABEL : B1

SEC : TSCRT  
TY : ORG  
LABEL : I5  
LABEL : H5  
LABEL : I3  
LABEL : B1  
LABEL : D5  
LABEL : C2  
LABEL : C4  
LABEL : E5  
LABEL : B2  
LABEL : G2  
LABEL : D2  
LABEL : A2  
LABEL : FRA5  
LABEL : DUD1  
LABEL : SUP3  
LEVEL : 3

Total number of records: 7  
Delete the retrieved record file: Y

**Step 11—Administrative personnel change the access rights of a user to include rights to intelligence data, but without authority to see FS (force status) records.**

**\*UPDATE**

REQUEST NAME: S11C1  
ACCESS TYPE: US  
FILE NAME: SCRTY  
USER TEMPLATE: DBUSE  
QUERY: USID=1111\*

**\*STATUS**

REQUEST NAME: S11C1

RECORD COUNT: 1

**\*EXAMINE**

REQUEST NAME: S11C1

DBA : Y  
USID : 1111  
FILE : TRACK  
AC : R  
QUERY : SEC=TSCRT/SEC=INTEL/TY=NSOF/TY=STAFF\*  
UT : CNUSE  
UT : CNORG  
UT : CNASC  
FILE : MESAG  
AC : R  
QUERY : SEC=TSCRT/SEC=INTEL/KIND=NSOF\*  
UT : CNMES  
AC : US  
QUERY : SEC=TSCRT/SEC=INTEL/KIND=NSOF\*  
UT : CNMES  
AC : AS  
QUERY : SEC=TSCRT/SEC=INTEL/KIND=NSOF\*  
UT : CNMES  
AC : D  
QUERY : SEC=TSCRT/SEC=INTEL/KIND=NSOF\*  
UT : CNMES

UPDATE? Y

INSERT? N

ATTRIBUTE: DBA  
VALUE: Y  
REPLACE(R), DELETE(D), OR NO CHANGE(CR)?  
INSERT? N

The access rights for a user are stored in a record in the file "SCR~~TY~~," each user having a separate record. These records are constructed, stored, retrieved, and updated just like any other records in HSDMS. Of course, special access rights are required to access the SCR~~TY~~ file. Here, an administrative user indicates that he wishes to update the record for "USID=1111" (this is the user ID of a particular user).

Here, the record is displayed, showing access control data in the form of attribute-value pairs, as with all records on HSDMS.

Under each file name is a list of the various access types permitted that user for that file (for TRACK, the only access type is "read"; for MESAG, the access types are "read," "update," "add," and "delete"). For each access type, there is a list of the user templates ("UT") which may be used with that access type, and a "QUERY." The QUERY is a Boolean expression of security keywords which specifies those records which the user is not allowed to access using that access type. In this case, the user is not allowed to see top secret, intelligence, or NSOF records in the MESAG file.

ATTRIBUTE: USID

VALUE: 1111

REPLACE(R), DELETE(D), OR NO CHANGE(CR)?

INSERT? N

ATTRIBUTE: FILE

VALUE: CONAN

REPLACE (R), DELETE(D), OR NO CHANGE(CR)?

INSERT? N

ATTRIBUTE: AC

VALUE : R

REPLACE(R), DELETE(D), OR NO CHANGE(CR)?

INSERT? N

ATTRIBUTE: QUERY

VALUE: SEC=TSCRT/SEC=INTEL/TY=NSOF/TY=STAFF\*

REPLACE(R), DELETE(D), OR NO CHANGE(CR)? R

REPLACEMENT VALUE: TY=NSOF/TY=STAFF\*

INSERT? N

ATTRIBUTE:UT

VALUE:CNUSE

REPLACE(R), DELETE(D), OR NO CHANGE(CR)?

INSERT? N

ATTRIBUTE:QUERY

VALUE:SEC=TSCRT/SEC=INTEL/KIND=NSOF\*

REPLACE(R), DELETE(D), OR NO CHANGE(CR)?

INSERT? N

ATTRIBUTE: UT

VALUE: CNMES

REPLACE(R), DELETE(D), OR NO CHANGE(CR)?

INSERT? N

Valid as entered? Y

DBA : Y

USID : 1111

FILE : TRACK

AC : R

QUERY : SEC=TSCRT/SEC=INTEL/TY=NSOF/TY=STAFF\*

REPLACEMENT VALUE : TY=NSOF/TY=STAFF\*

UT : CNUSE

Here, the user has told the system he wishes to update the record. The system prompts the user for each attribute in the record to see if the user wants to replace or delete the value. The system also prompts the user between each attribute ("INSERT?") to see if the user wants to insert a new attribute-value pair between existing ones. Once all attributes have been processed, the system requests the user to verify that all data entered is valid.

Finally, the system displays the entire record, along with the changes made, and asks for final confirmation before actually updating the record in the database.

MANOLA AND HSIAO

UT : CNORG  
UT : CNASC  
FILE : MESAG  
AC : R  
QUERY : SEC=TSCRT/SEC=INTEL/KIND=NSOF\*  
UT : CNMES  
AC : US  
QUERY : SEC=TSCRT/SEC=INTEL/KIND=NSOF\*  
UT : CNMES  
AC : AS  
QUERY : SEC=TSCRT/SEC=INTEL/KIND=NSOF\*  
UT : CNMES  
AC : D  
QUERY : SEC=TSCRT/SEC=INTEL/KIND=NSOF\*  
UT : CNMES

In this case, the QUERY for reading records in the TRACK file has been altered so that top secret and intelligence records are no longer forbidden to the user.

Final update confirm: Y

Number of records updated: 1

Total number of records: 1